Niveau(x) d'enseignement: 4 Numéro EDUQ : 8870368
Centre d'intérêt: 846

St-Cyr, Pierre.

Introduction à la science de l'informatique, ISI 111-434, 111-534 : projet d'élèves / [Pierre Saint-Cyr] ; Direction générale des programmes, Direction de la formation générale. -- [Québec] : Ministère de l'éducation, c1988.
47 p. : ill. ; 28 cm. -- (Document d'information / Ministère de l'éducation du Québec, Direction générale des programmes)

Code: 16-0081-03 ; 8889-103. Bibliogr.: p. 47.

ISBN 2-550-13996-8

Vedette(s)-secondaire(s): Québec (Province) Ministère de l'éducation. Direction de la formation générale

Descripteur(s): Initiation à l'informatique * Projet d'étudiant * Elaboration de projets * Guide pédagogique * Secondaire (Ordre d'enseignement)

Identificateur(s) de sujet: Introduction à la science de l'informatique

Le programme "Introduction à la science de l'informatique", connu également sous le sigle ISI, valorise l'enseignement par projet. Cet enseignement par projet pose des problèmes d'encadrement pour les enseignants peu familiers avec une telle méthode. Ce guide propose une façon de diriger des projets où l'enseignant, contrairement à son rôle habituel, doit aider l'élève à structurer et à organiser son travail. Il s'agit d'amener l'élève à réfléchir sur le papier avant d'entreprendre le travail sur l'ordinateur. Le document comprend quatre parties: 1. une démarche pour ISI; des activités précises s'inscrivent à l'intérieur de cette démarche pédagogique: exploration-découverte, acquisitions d'habiletés et objectivation; 2. les étapes d'élaboration d'un projet; 3. le calendrier des activités; 4. les arbres algorithmiques: on présente les différents schémas informatiques fondamentaux sous la forme d'arbres algorithmiques.

Type de document: d
Notice(s) connexe(s): 8770660

EDUQ 8870368

# DOCUMENT D'INFORMATION

## INTRODUCTION À LA SCIENCE DE L'INFORMATIQUE ISI 111-434, 111-534

### PROJET D'ÉLÈVES

Québec

EDUQ 8870368

# INTRODUCTION À LA SCIENCE DE L'INFORMATIQUE ISI 111-434, 111-534

## PROJET D'ÉLÈVES

**Direction générale des programmes**
Direction de la formation générale

© Gouvernement du Québec
Ministère de l'Éducation, 1988

ISBN 2-550-13996-8

Dépôt légal - deuxième trimestre 1988
Bibliothèque nationale du Québec

# I.S.I. 111-434 111-534


Introduction..... 3

Étapes d'élaboration... 11

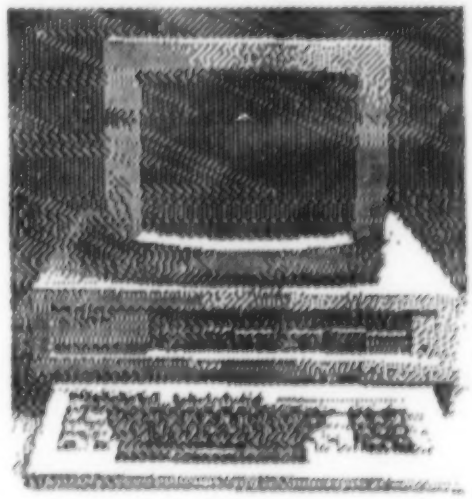
Planification ..... 25

•Enseignement

•Projet

Arbres algorithmiques.. 37


## Introduction à la Science de l'Informatique.




Pierre Saint-Cyr,
prof. d'I.S.I.

## PRÉSENTATION

Le programme d'études Introduction à la Science de l'Informatique, connu également sous le signe ISI, est un programme innovateur qui a réussi à s'implanter avec succès dans le milieu scolaire, malgré les nombreuses difficultés tant au niveau du contenu et de l'approche d'ISI, qu'au niveau des modalités d'implantation.

Dans le long processus d'application d'un programme (comme ISI) jusque dans la classe, il devient nécessaire à certains moments critiques de soutenir et d'aider les responsables premiers de la vie scolaire, les enseignantes et enseignants, dans leur intervention pédagogique auprès des élèves. C'est ainsi que le Guide pédagogique venait dans un premier temps appuyer le Programme; ce Guide donnait des indications générales afin d'éviter que l'apprentissage et l'enseignement dans le cadre d'ISI deviennent prématurément «figés dans le béton», et trop formels. Cette stratégie prévoyait également l'ajout de documents d'information portant sur des « objets » présentant des difficultés particulières, objets identifiés à partir des évaluations régulières des phases d'implantation.

C'est ainsi que trois thèmes ont été retenus; chacun de ceux-ci donnera lieu à une publication précise dont la fonction sera d'aider à la relation maître-élève en fournissant aux intervenants des indications quant à des façons de faire et à des points particuliers dont il est utile de tenir compte. Les thèmes qui seront traités sont les suivants: LE PROJET D'ÉLÈVES, L'ÉVALUATION et LA RÉSOLUTION DE PROBLÈMES.

Le présent document d'information concerne le PROJET D'ÉLÈVES et s'inscrit dans cette démarche de soutien et d'aide qui doit avoir un caractère de souplesse et de respect. Il est à espérer qu'il pourra contribuer à une éducation de qualité.

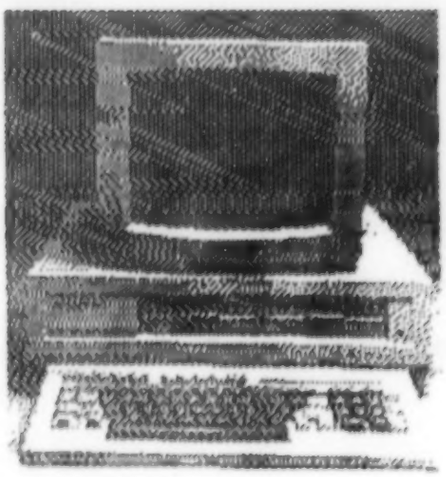
NOTE: Tout au long de ce document le générique masculin sera utilisé sans aucune discrimination et uniquement dans le but d'alléger le texte.

Bertrand A. Morin
Ministère de l'Education

# I.S.I.

Introduction.....p. 3-8

## Une démarche pour ISI



## Objectif général

Le programme Introduction à la Science de l'informatique vise à développer, chez les étudiants, une pensée logique et structurée. Le cours n'est donc pas, en tant que tel, un cours de programmation. Une approche axée sur l'acquisition d'une méthode de travail est suggérée. La programmation n'est que la concrétisation d'un processus logique, d'un algorithme précis. Réaliser un projet consiste à apprendre à chercher de l'information et à organiser cette information. Le projet-étudiant devient alors une démarche importante, complète et personnelle vers l'autonomie. Contrairement à son rôle habituel de dispensateur de cours, l'enseignant ou l'enseignante aide l'élève à structurer et à organiser son travail. Il s'agit d'amener les étudiants à "réfléchir sur papier" avant d'entreprendre le travail sur ordinateur. Comme le souligne le guide pédagogique: " (....) l'initiation ne doit pas se faire à un langage mais à un ensemble de concepts et de méthodes qui s'incarnent à travers l'utilisation que l'on fait du langage." ( cf. p. 7 )

## Une démarche...

Des activités précises s'inscrivent à l'intérieur d'une démarche pédagogique:

exploration-découverte, acquisitions d'habiletés et objectivation.

## Exploration-découverte

- <u>consulter des projets des années antérieures</u>

- charger le logiciel;
- le faire exécuter;
- analyser de façon critique les divers éléments: présentation, structure, documentation, . . .

• consulter un manuel de référence du langage utilisé

• consulter des documents d'appoint sur l'utilisation de procédures ou de sous-programmes

• compléter des exercices d'exploration-découverte qui vont permettre:

- d'identifier les différentes structures: séquentielles, répétitives et branchements conditionnels;
- de découvrir les erreurs de syntaxe de même que les erreurs de logique.

• élaborer une approche personnelle de résolution de problèmes

**Acquisition d'habiletés**

Les premiers projets-étudiants ont pour objectif de développer des habiletés précises qui seront par la suite réinvesties au niveau de projets plus complexes. C'est ainsi que les projets pourraient avoir pour thèmes les points d'intérêts suivants:

- afficher des textes;
- faire des dessins;
- faire des calculs;
- faire de la musique.

Il va sans dire que cette étape est fortement marquée par l'intervention de l'enseignant ou de l'enseignante qui suggèrent des façons de faire et des façons de s'organiser. C'est la période pendant laquelle les élèves vont développer une bonne méthode de travail.

**Objectivation**

Une bonne connaissance de certains concepts fondamentaux, la pratique de techniques d'analyse descendante ainsi que les habiletés à élaborer de bons algorithmes vont permettre d'élaborer un projet. Les outils développés antérieurement sont réinvestis au niveau d'une création personnelle.

**L'analyste-programmeur**

Les objectifs généraux 1 et 3 ainsi que les objectifs afférents soulignent de façon précise le travail de l'analyste-programmeur. Ce travail met en évidence les étapes d'élaboration de projets et s'inspire des techniques de conception de logiciels (en anglais, software design).

**La résolution de problèmes**

L'objectif général 2 et les objectifs terminaux et intermédiaires rendent compte d'une réalité beaucoup plus vaste et plus difficile à circonscrire dans un domaine bien précis. Ces objectifs relèvent plutôt de la méthode de travail. Les outils intellectuels ainsi développés sont plus souvent qu'autrement des techniques ou des méthodes. C'est ainsi qu'une méthode de resolution de problèmes proposera de préciser le problème, de rechercher un problème plus simple, de décomposer le problème, de retourner au problème initial et de généraliser le problème en l'incluant dans un problème plus vaste.

## Les heuristiques

Cependant, pour en arriver à développer un ou plusieurs algorithmes de résolution d'un problème, un savoir-faire indispensable intervient, ce savoir-faire consiste en un ensemble de méthodes diverses, souvent définies de façon partielle et qui agissent selon un processus plus ou moins évident - selon les circonstances et les cas particuliers; ce sont les heuristiques. Ces "mécanismes" sont mis en place lors des phases d'exploration-découverte et d'acquisition d'habiletés. L'enseignant ou l'enseignante interviendront judicieusement en amenant l'élève à la découverte de la solution plutôt que par un enseignement direct de cette solution; c'est ici que l'appel aux heuristiques joue un rôle fondamental dans l'apprentissage. Ces méthodes sont nécessaires lors de la simulation d'un raisonnement et permettent ainsi de restreindre l'espace des recherches aux directions les plus vraisemblables. Voici quelques-uns de ces " savoir-faire ":

- chercher l'information nécessaire;
- identifier des sources d'information (banque de données, livres, revues, ...);
- sélectionner les informations utiles et essentielles;
- organiser l'information pour la rendre utilisable;
- développer et utiliser des stratégies de résolution de problèmes;
- savoir interpréter des messages d'erreurs et découvrir les causes immédiates de telles situations.

## Le projet formatif

La démarche pédagogique proposée précédemment amène l'élève à élaborer un scénario qui permet de comprendre certains concepts, de développer certaines habiletés et de créer un projet personnel.

Le projet formatif sert de prétexte et favorise l'action auprès des élèves en vue d'assurer la meilleure progression des apprentissages. L'enseignant ou l'enseignante présentent un projet élaboré dans toutes ses étapes de création. Chaque étape du projet formatif propose des techniques d'analyse et des exercices d'apprentissage.

La planification du projet formatif tient compte de trois aspects bien précis:

**. Fixer le but à atteindre**

-connaître certains concepts (programme, procédure, projet,...);
- connaître les structures (séquentielles, répétitives et les branchements conditionnels);
- développer certaines habiletés (manipuler des textes, faire des calculs, ...);
- déterminer si le nouvel apprentissage repose sur des préalables;
- déterminer les exercices d'exploration-découverte pour développer ces apprentissages

**. Déterminer les moyens**

- connaître à l'avance les problèmes, les causes, les solutions et les actions à entreprendre.

**. Préciser le scénario d'utilisation**

- se situer dans une approche de pédagogie ouverte;
- analyser les apprentissages susceptibles d'intéresser les élèves ( graphisme, affichage, animation, . . .).

## Un projet coopératif

Ce projet formatif va servir d'élément déclencheur à toutes sortes d'initiatives. Si le sujet est intéressant et bien présenté, il saura motiver les élèves. Dans certains cas l'approche proposée favorise l'élaboration d'un projet coopératif. Selon les intérêts de chacun pour le graphisme, la mise en page, etc., les diverses équipes travaillent sur une partie spécifique du grand projet. Le travail se matérialise sous la forme d'une unité de programme autonome. La mise en commun des parties constituantes du projet contribue à l'enrichissement collectif et permet des échanges très intéressants au niveau du travail de chaque équipe de production.

## En résumé

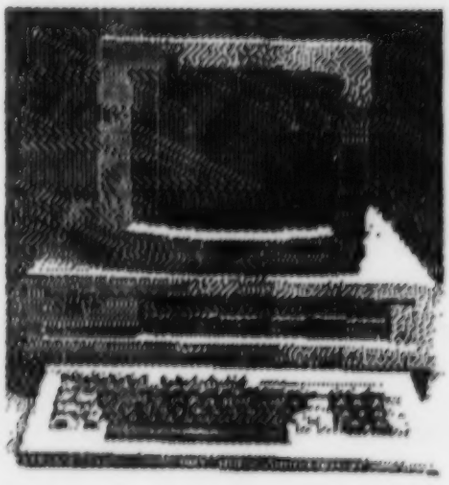
Un projet comme celui-ci permet de dégager:

- le concept de programme;
- l'utilité et l'utilisation des procédures;
- l'utilisation des structures informatiques;
- la résolution de problèmes.

# I.S.I.

**Étapes d'élaboration....p.11- 21**

## Les étapes d'élaboration



## Les étapes d'élaboration de projets

Nous allons, dans les pages qui suivent, voir les étapes d'élaboration du projet. Par la suite, nous tenterons de replacer ces mêmes étapes dans un cadre temporel.

Programme d'Études, p. 23 (Objectifs correspondant aux étapes)

1. L'énoncé du projet

Obj. 1.1: CONSTRUIRE un énoncé clair d'un projet issu d'expériences du quotidien.

2. La définition du projet

Obj. 1.2: REFORMULER un énoncé clair du projet retenu.

3. L'analyse du projet

Obj. 1.3: ORGANISER le travail par un processus permettant de passer de l'information initiale à l'information finale désirée.

4. La spécification

Obj. 1.4: DÉCRIRE le processus de façon systématique en tenant compte des possibilités du langage de programmation et de l'ordinateur.

5. La réalisation

Obj. 1.5: TRADUIRE ce processus en un algorithme codable.

Obj. 1.6: CODER les actions en instructions de programme.

Obj. 1.7: EXPLOITER (ou mettre en application) le programme sur ordinateur.

6. Les tests

Obj. 1.8: METTRE AU POINT le programme ( évaluer, épurer, débugger).
1.8.1 à 1.8.8
1.8.13 à 1.8.19

7. L'entretien

Obj. 1.8: METTRE AU POINT le programme ( évaluer, épurer, débugger).
1.8.9 à 1.8.12

## 1. L'énoncé du projet

La première étape est l'énonciation du projet à réaliser: "Je veux faire un petit tutoriel de mathématiques pour les élèves de l'élémentaire."

Voici une démarche susceptible de guider l'élève dans le choix d'un projet précis:

- faire une liste de ses goûts personnels;
- consulter une liste de projets:
  - revues d'informatique;
  - livres d'informatique ( Projets et Programmes, Annexe, p. 233; A vous de jouer, Troisième partie: Ce que l'ordinateur peut faire... avec vous, p. 177);
- rechercher les ressources existantes:
  - projets existants qui simulent un aspect du même projet;
  - littérature qui traite d'un sujet particulier;

• consulter des exemples de projets documentés:
( **projets formatifs** disponibles en version BASIC ou LOGO au Centre de Développement de la Micro Informatique, Commission des Ecoles Catholiques de Montréal)
- cahier pour l'étudiant;
- guide méthodologique pour l'enseignante ou l'enseignant;
- logiciels sur disquettes en version BASIC et LOGO.

• charger en mémoire et faire exécuter des projets des années antérieures:
- banque de projets sur disquettes ( échange avec d'autres écoles ou d'autres régions du Québec);

• consulter l'enseignant et ses amis:
- échanger, discuter avec les membres des autres équipes pour se faire une idée bien précise du projet que l'on a en tête;
- consulter l' enseignant sur la faisabilité de ce projet (un projet trop difficile risque d'être abandonné, tandis qu'un projet trop facile ne présente aucun défi particulier);

• faire son choix:
- l'idée du tutoriel de mathématiques semble être intéressante;
- le degré de difficulté correspond aux possibilités des membres de l'équipe.

## 2. La définition du projet

La bonne définition d'un projet commence par une simulation écrite. Cet exercice d'écriture permet de dégager les grandes étapes du projet. Voici les grandes lignes directrices du tutoriel de mathématiques:

- l'élève de l'élémentaire aura à choisir entre faire des additions, des soustractions, des multiplications, des divisions, des opérations mixtes ou, s'il le désire, quitter le programme;

- chaque choix comporte 10 problèmes à résoudre;

- à chaque réponse il y aura:
  - un message de félicitations si la réponse est exacte;
  - dans le cas contraire, l'ordinateur donne la bonne réponse;

- le total de bonnes réponses est affiché pour chaque série de calculs;

- tant que l'élève ne choisira pas de quitter le programme, il aura la possibilité de faire une nouvelle série de 10 problèmes.

Nous avons là une première formulation en langage courant. Ceci permet de voir clairement les choses à faire:

- les grandes étapes du projet à réaliser et les problèmes à résoudre;

- éviter les réponses négatives dans la soustraction;

- présenter des divisions sans reste.

**En résumé:**

Pour définir le projet, il suffit de faire une simulation écrite et établir un scénario précisant:

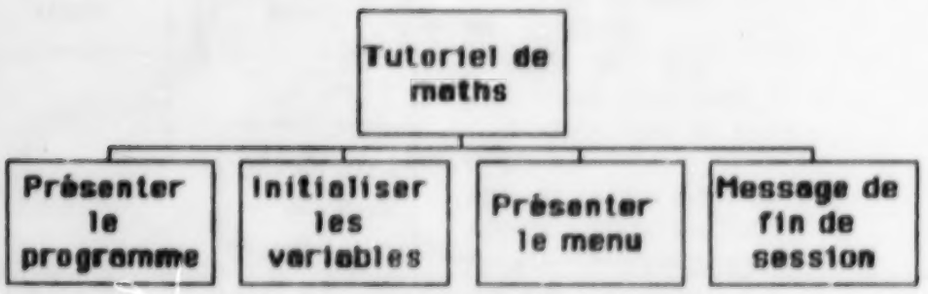
- les entrées: quelles sont les choses que l'ordinateur doit connaître pour passer à l'exécution ?
- le traitement de l'information: qu'est-ce qui doit être fait ?
- les sorties attendues: quel sera le produit final ou le but désiré ?

## 3. L'analyse

L'analyse va permettre de faire une première formulation générale du projet à réaliser:

- dégager et définir les grandes étapes du projet:
  - faire l'arbre algorithmique du projet;
  - écrire en pseudo-langage les grandes étapes.

**Faire l'arbre algorithmique (analyse descendante)**




**Décrire chaque tâche en pseudo-langage**

**Programme Tutoriel**

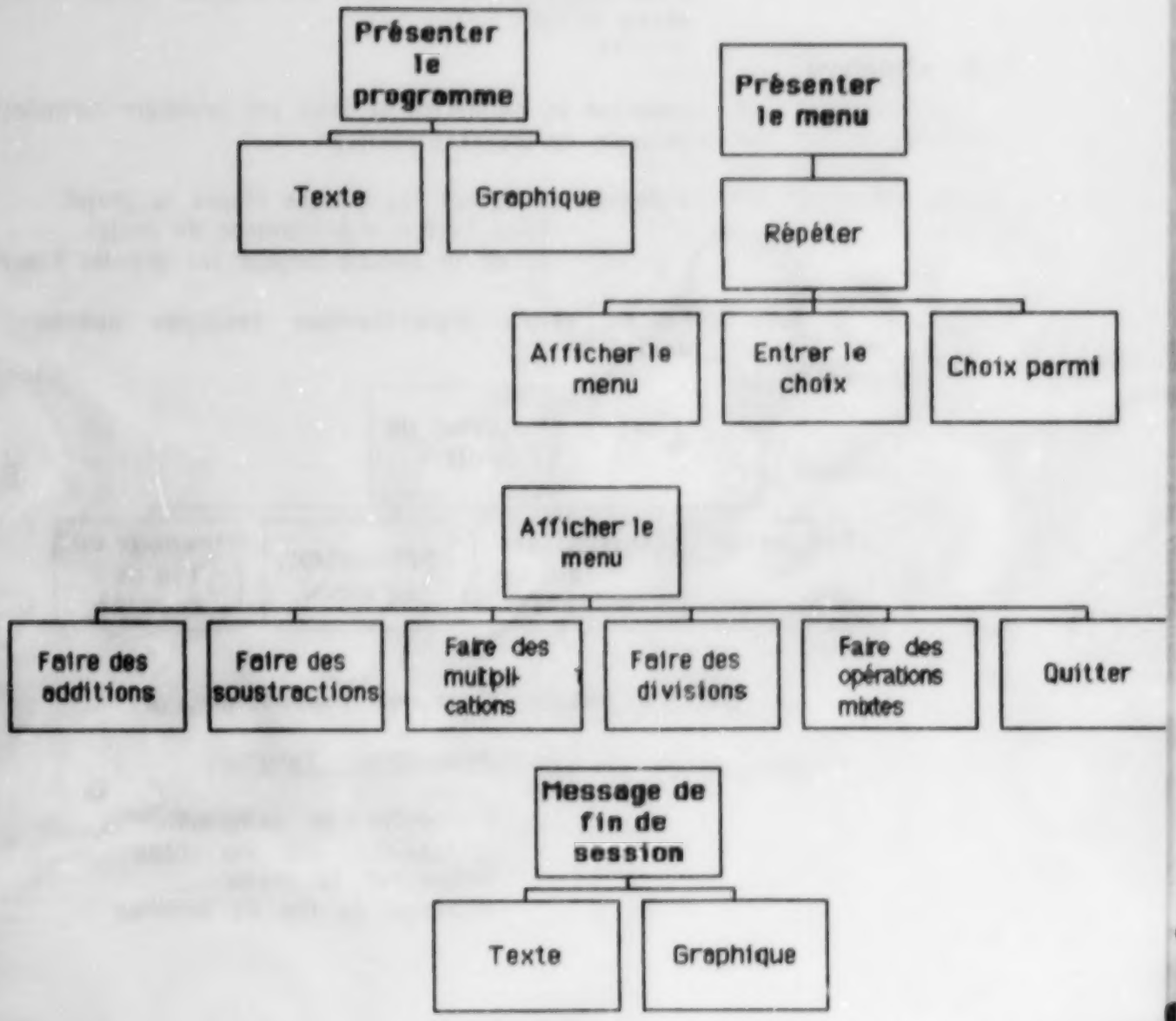
**Présenter le programme**
**Initialiser les variables**
**Présenter le menu**
**Message de fin de session**

## 4. La spécification

La spécification permet de reformuler les différentes parties du problème.

- reformuler les grandes étapes du projet
  - raffiner l'arbre algorithmique
  - raffiner les étapes du pseudo-langage.

**L'arbre algorithmique**



## Le pseudo-langage

Une formulation quasi complète pourrait être la suivante:

**•Présenter le programme**
Texte;
Graphique;

**•Initialiser les variables**

**•Présenter le menu:**
Répéter:
-Afficher le menu:
Faire des additions;
Faire des soustractions;
Faire des multiplications;
Faire des divisions;
Faire des opérations mixtes;
Quitter;
-Entrer le choix:
Ecrire un message;
Lire le choix:
Ecrire ("_____");
Lire (Choix);
-Valider le choix:
Si choix de 1 à 6:
alors continuer;
sinon lire le choix;
-Choix parmi:
Appeler ADDITION;
Appeler SOUSTRACTION;
Appeler MULTIPLICATION;
Appeler DIVISION;
Appeler MIXTE;
Jusqu'à choix = quitter;

**•Message de fin de session:**
-Texte;
-Graphique.

## 5. La réalisation

La réalisation consiste à coder les modules dans un langage de haut niveau.

### La documentation-programme

Le fait d'utiliser BASIC, entre autres, pour coder un algorithme pourrait signifier un programme parfaitement incompréhensible pour un non-initié. Pour remédier à cet état de choses:

- il faut bien structurer le travail à coder;
- il faut bien documenter les différentes étapes à même le listage du programme.

Voici deux exemples de programme:

**Salaire des employés**

```
DÉBUT
     Appeler "Présentation_programme"
     Appeler "Lecture"
     Tant que No<>0
          Appeler "Calcul du salaire"
          Appeler "Afficher_salaire"
          Appeler "Lecture"
     FinTant
     Appeler   "Afficher_salaire_total"
     Appeler "Afficher_message_de_la_fin"
FIN
```

**Imprimer un texte à l'écran**

```
Procédure  Mots
Procédure  Imprime

DÉBUT
     Appeler   Mots
     Appeler   Imprime
FIN
```

Codage BASIC

```
10 'programme salaire des employés
15 gosub 1000 'Présentation_programme
20 gosub 2000 'Lecture
25 while NO<>0
30          gosub 3000 'Calcul _du_salaire
35          gosub 4000 'Afficher_salaire
40          gosub 2000 'Lecture
45 wend
50 gosub 5000 'Afficher_salaire_total
55 gosub 6000 'Afficher_message_de_la_fin
60 'Les procédures
1000 'Présentation_programme
1010 ...
1020 ...
...
1900 return
2000 'Lecture
...
```

Codage PASCAL

```
program Texte;
 var
  mot1, mot2, mot3 : string[40];
 procedure Mots;
 begin
  mot1 := 'L' ordinateur peut
          aussi bien';
  mot2 := 'produire un nombre
          aléatoire';
  mot3 := 'qu une liste de nombres
          aléatoires';
 end;
 procedure Imprime;
 begin
   writeln(mot1);
   writeln(mot2);
   writeln(mot3);
 end;
begin
 Mots;
 Imprime;
end.
```

### La documentation-utilisateur

Un texte approprié présente le logiciel à l'utilisateur éventuel. Une présentation claire et précise donne le mode d'emploi du programme et explique les diverses fonctions. Des explications supplémentaires sont disponibles sur demande. Dans certains cas, un guide d'accompagnement est fourni avec le produit final.

## 6. Tester le programme

Un programme informatique est rarement exécuté du premier coup sans problèmes. Il faut vérifier si le programme fonctionne bien dans tous les cas possibles.

<u>Il faut faire un plan de tests:</u>

- avoir une stratégie de mise au point:
  - vérifier avec des données valides;
  - établir des mesures préventives;
  - vérifier les cas limites;

- vérifier les formats de sortie;

• faire un jeu d'essais: fournir des ensembles de données qui vérifient le fonctionnement global du programme;

• faire la simulation sur papier en établissant la trace des variables;

• utiliser des outils de mise au point:
  BASIC: TRON, TROFF, ON ERROR GOTO.....
  LOGO: ATTRAPE "ERREUR
  PASCAL: lors de la compilation d'un programme.

**7. L'entretien:** Après une certaine période de rodage, il n'est pas rare de vouloir apporter des améliorations à un programme. En effet, l'utilisation qu'on en fait, les gens qui l'utilisent et les commentaires que nous recevons nous amènent à modifier certaines choses. Si le programme est bien construit et bien structuré, il sera facile:

• de corriger ou de changer les parties qui ne fonctionnent pas;

• d'éliminer les parties inutiles;

• d'ajouter de nouvelles parties qui:
  - augmentent l'efficacité du programme;
  - ajoutent des fonctions nouvelles.

**En résumé:**

Faire un projet consiste à:

- Préciser le problème à résoudre en décrivant les données du problème et les résultats attendus;

- Décomposer l'action globale que doit réaliser le programme en actions plus élémentaires (démarche descendante);

- Décrire le fonctionnement de chaque partie en utilisant un pseudo-code lisible par tout le monde;

- Décider des variables à utiliser en choisissant leur nom de façon imagée et en distinguant leur nature (globale, locale, paramétrisée,...);

- Rédiger chaque partie dans le langage choisi en lui donnant un titre correspondant à l'action attendue et en indiquant le sens et la nature de chaque variable. On commence par les programmes élémentaires et on "remonte" vers le plus complexe (programmation ascendante): on couvre ainsi l'ensemble;

- Indiquer en quoi l'exécution d'une partie de programme qui vient d'être écrite modifie l'état global du programme. Cet état est constitué des valeurs, des variables, des paramètres et des constantes utilisés dans le programme complet à un instant donné;

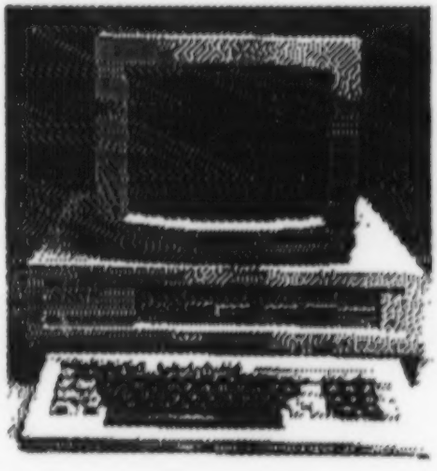
- Tester chaque partie au fur et à mesure, en commençant par les parties ne faisant appel à aucun autre programme (programmes élémentaires).

# I.S.I.

**Planification** .....p. 25-33

- **Enseignement**
- **Projet**

## Le calendrier des activités



## Utilisation d'un logiciel de gestion.

Il existe des logiciels de gestion de projets qui peuvent être, dans la majorité des cas, des outils intéressants pour la planification de projets. Prenons ici le mot "projet" dans son sens large. L'enseignant peut se servir d'un tel logiciel pour:

- planifier ses interventions auprès des élèves;
- planifier les étapes d'élaboration d'un projet.

Pour utiliser un tel logiciel l'enseignant doit:

1. définir le projet;
2. élaborer un calendrier des activités:
   - activités préalables;
   - activités parallèles;
3. planifier l'horaire du projet:
   - durée d'une activité;
   - date du début de l'activité;
   - date de complétion.

## Outils de gestion d'un projet.

Par exemple, ces logiciels permettent le traitement du projet:

- Mac Project (logiciel utilisé sur Macintosh);
- HPM (Harvard Project Management utilisé sur IBM et compatibles).

**N.B.** Ces logiciels sont fondés sur des méthodes bien connues:

**P.E.R.T.** (Project Evaluation and Review Technics, 1950) Le programme permet de calculer les optimums d'un réseau de contraintes portant sur les tâches successives d'un cahier.

**C.P.M.** (Critical Path Method, 1957)

## Pourquoi utiliser un tel logiciel ?

Un logiciel de gestion permet de:

- visualiser les tâches d'un projet ainsi que les interrelations entre ces différentes tâches;
- calculer immédiatement les effets d'un changement quelconque dans la planification du projet;
- décider de la meilleure façon d'utiliser les ressources;
- communiquer aux différents intervenants les choses importantes à faire pour la réalisation du projet à l'intérieur d'une période donnée.

## Concepts de base

**•Projet:**

Le projet est un ensemble d'activités interreliées directement, orientées vers un but spécifique et devant être réalisées dans un cadre temporel spécifique.

**•Tâches (Task):**

La tâche est l'activité spécifique qui a une définition précise.

**•Jalon ( Milestone):**

Le jalon est un événement spécifique dans le temps, qui peut-être décrit objectivement.

**•Chemin critique:**

Le chemin critique est le trajet qui, en unissant l'ensemble des tâches représentées par un graphe, correspond au temps total le plus court pour l'exécution de cet ensemble de tâches. C'est le chemin dont le flottement est nul.

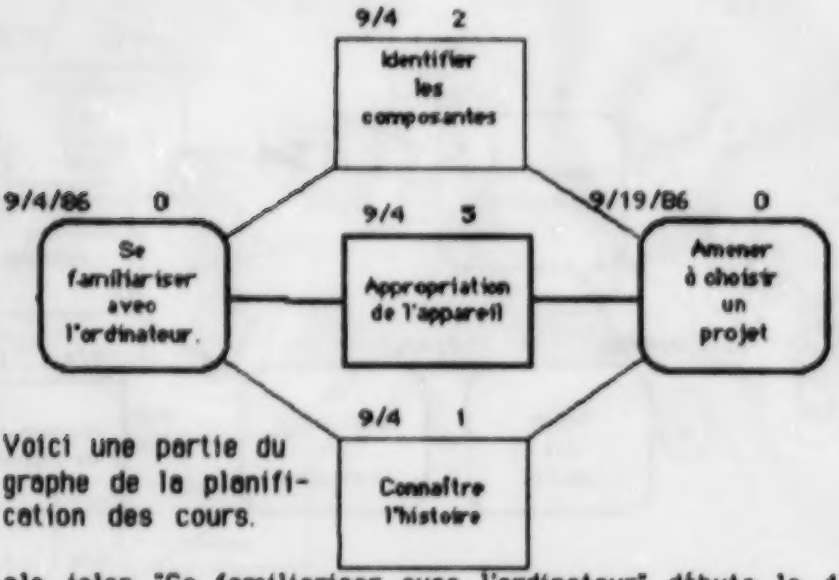
•**Flottement ou délai:**

Le délai est la différence entre la durée d'un chemin quelconque sur un graphe de tâches, et le chemin critique qui représente le temps le plus court possible pour l'exécution du projet qui réunit l'ensemble des tâches en cause. C'est en somme un indicateur de la marge de manoeuvre disponible.

•**Graphe:**

C'est un instrument de la gestion logique des ressources. L'allure en réseau du graphe est donnée par le fait que plusieurs séries de tâches peuvent se développer simultanément.

**Éléments d'un graphe**




Voici une partie du graphe de la planification des cours.

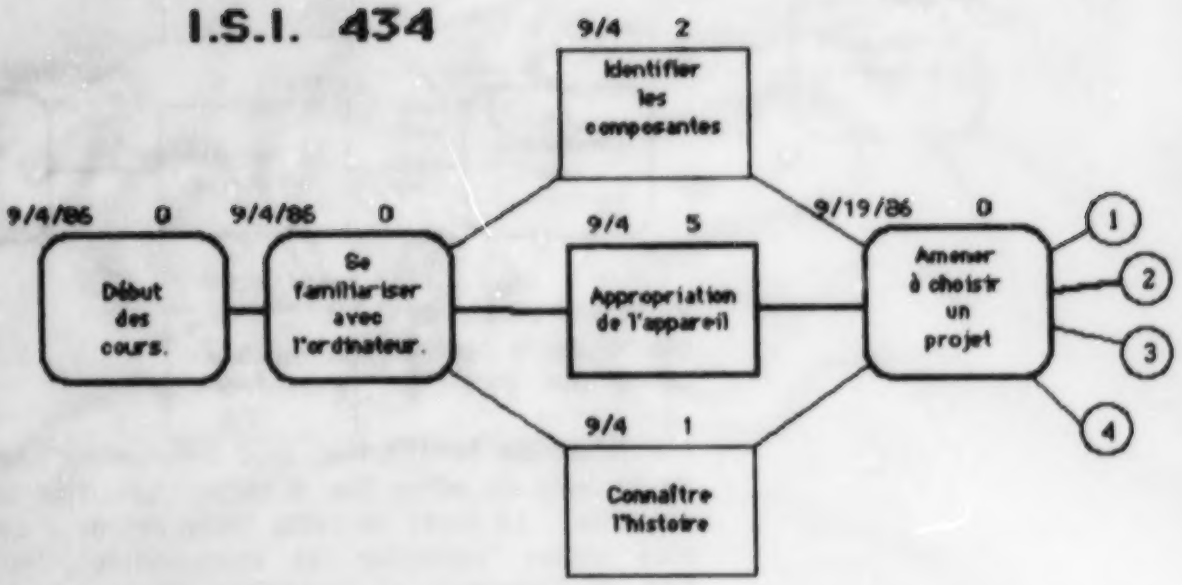
•le jalon "Se familiariser avec l'ordinateur" débute le 4 du 9e mois de même que la tâche "Identifier les composantes". La durée de cette tâche est de 2 cours.
•les tâches "Identifier les composantes", "Appropriation de l'appareil" et "Connaître l'histoire" s'effectuent parallèlement
•le trait en caractères gras entre "Se familiariser avec l'ordinateur", "Appropriation de l'appareil" et "Amener à choisir un projet" indique un chemin critique

## Un exemple de planification

Voici le graphe d'une planification de l'enseignement qui tient compte des facteurs mentionnés précédemment.

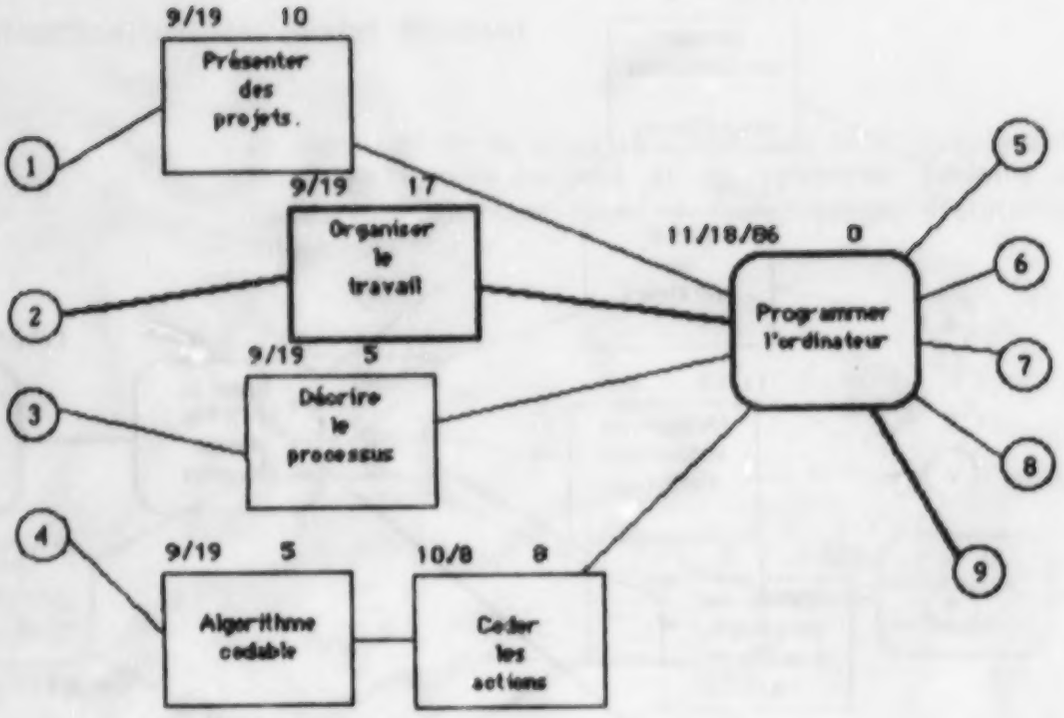
Familiarisation (du 4 sept. 1986 au 19 sept. 1986):

- identification des composantes (durée: 2 cours);
- appropriation de l'appareil (durée: 5 cours);
- historique des ordinateurs (durée: 1 cours);



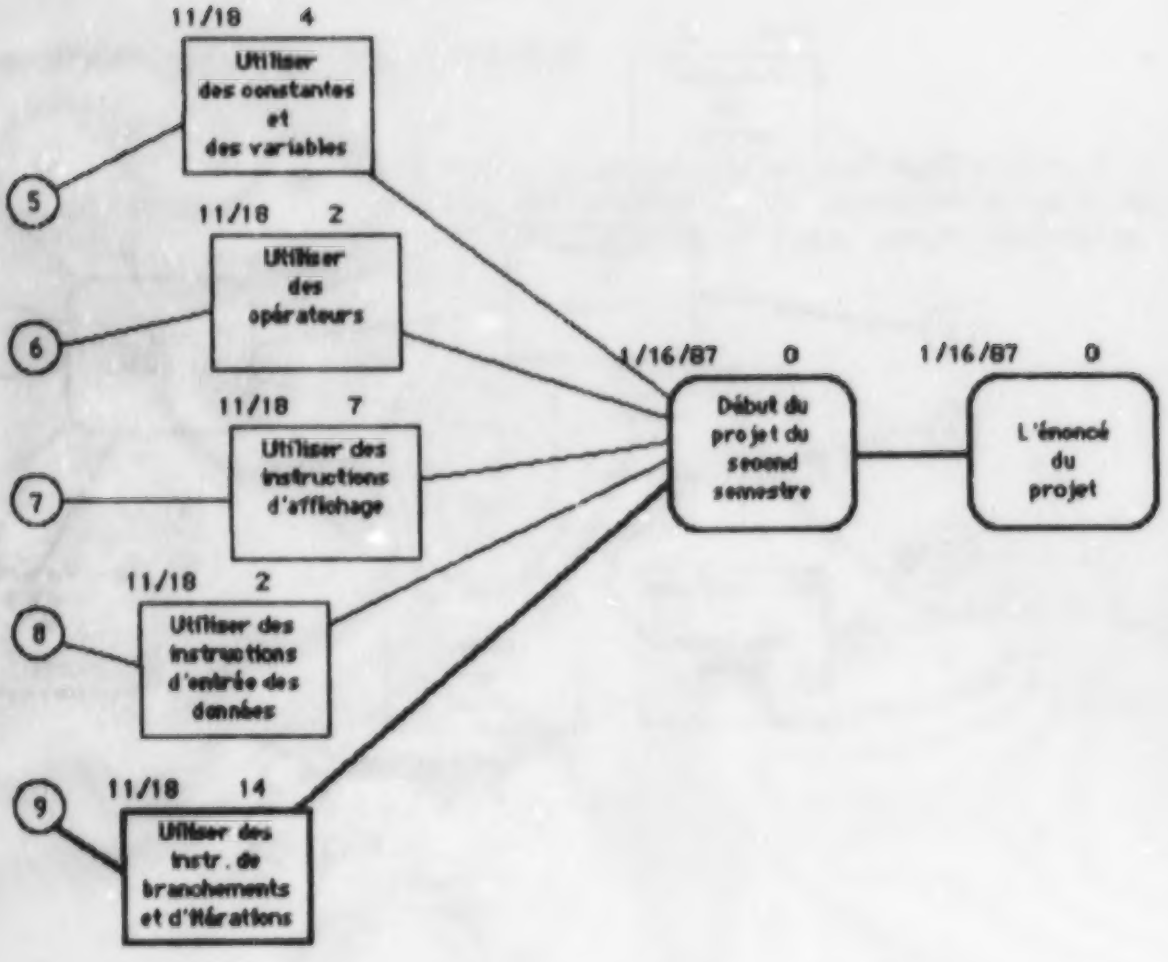
Choisir un projet (du 19 sept. 1986 au 18 nov. 1986):

- présentation de projets (durée: 10 cours);
- organisation du travail (durée: 17 cours);
- description des tâches (durée: 5 cours);
- élaboration d'un algorithme (durée: 13 cours);



Programmer (du 18 nov. 1986 au 16 janv. 1987):

- utilisation de constantes, de variables et d'opérateurs (durée: 4 cours);
- utilisation d'instructions d'affichage (durée: 2 cours);
- utilisation de structures de contrôle (durée: 14 cours);
- etc.

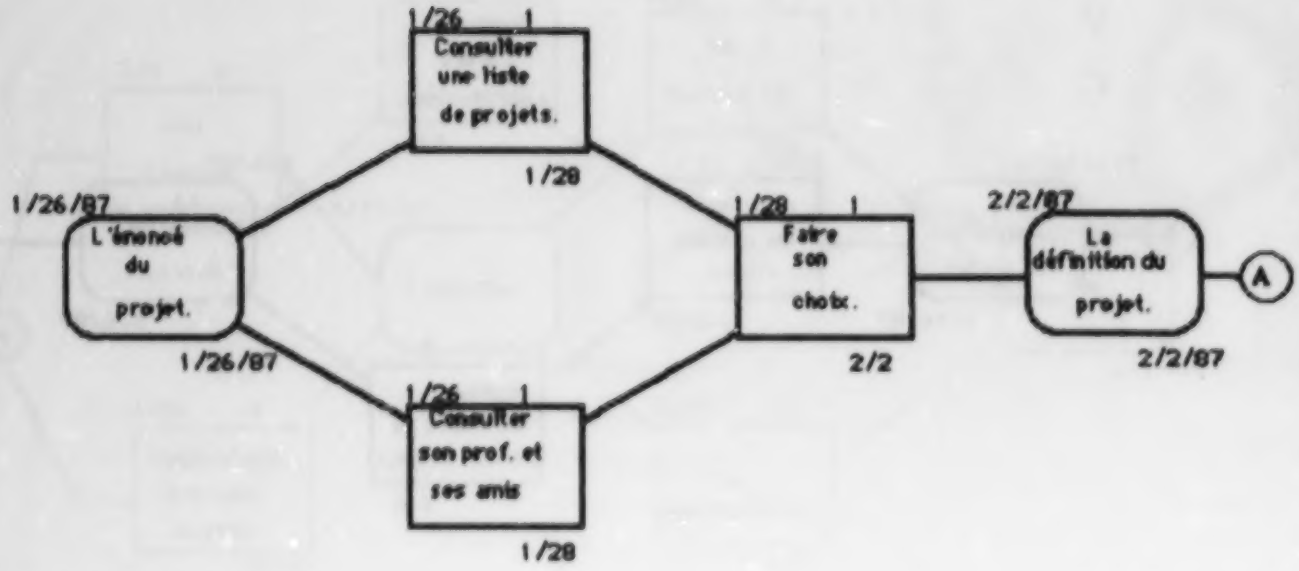


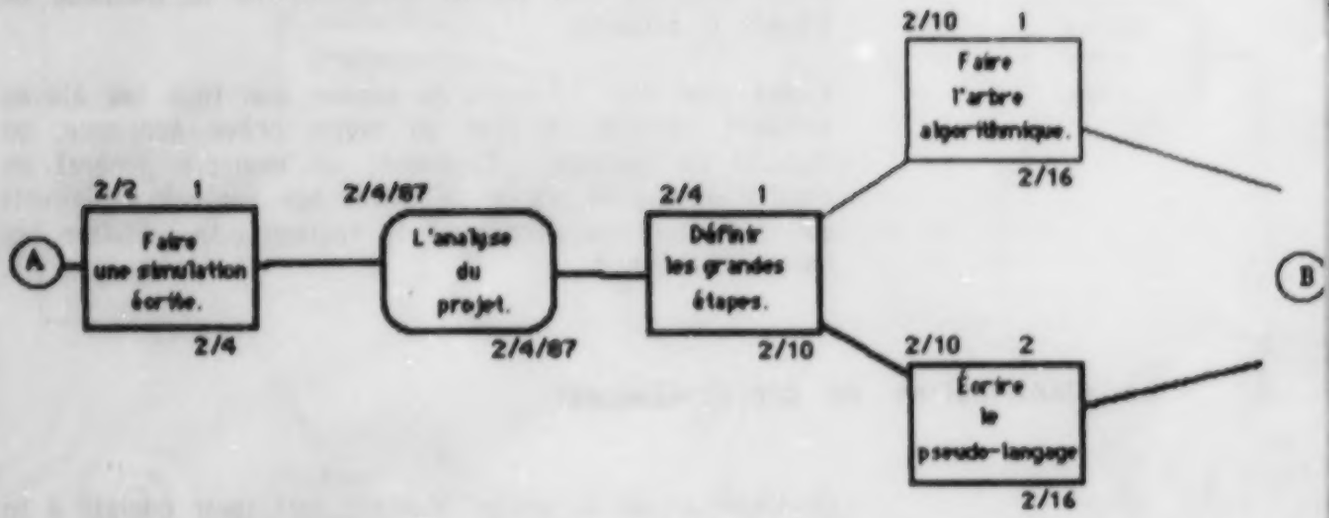
La planification des différentes étapes d'élaboration du projet-étudiant fait partie intégrante de la méthode de travail à acquérir.

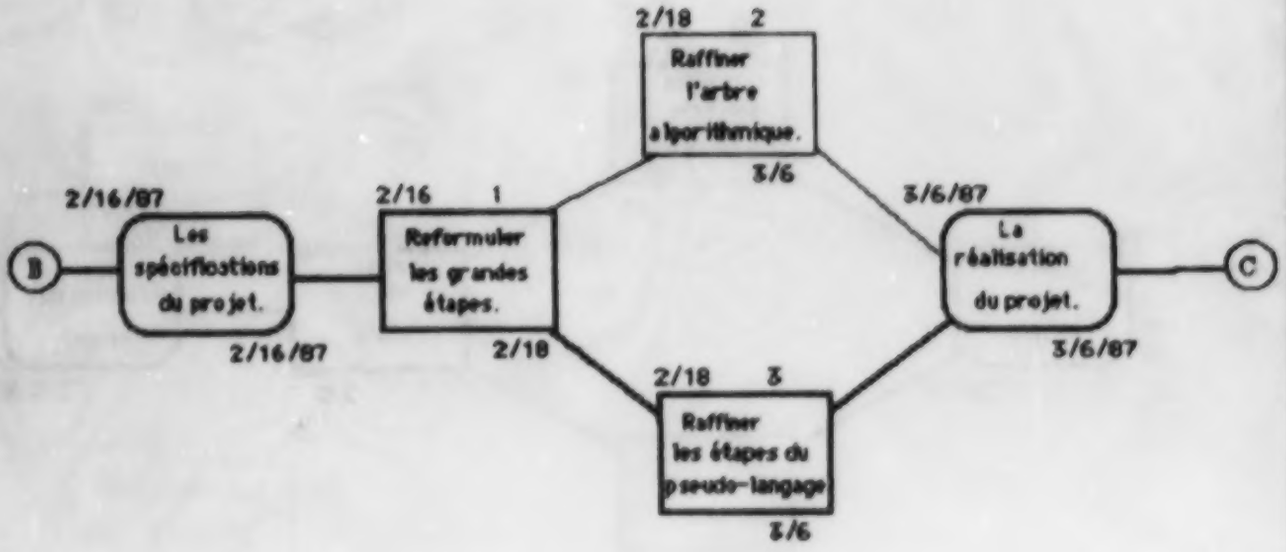
Il est peut-être illusoire de penser que tous les élèves puissent utiliser, à plus ou moins brève échéance, un logiciel de gestion. Cependant un exemple général de planification peut servir de balise aux équipes de travail qui consultent régulièrement le tableau pour vérifier les dates d'échéance.

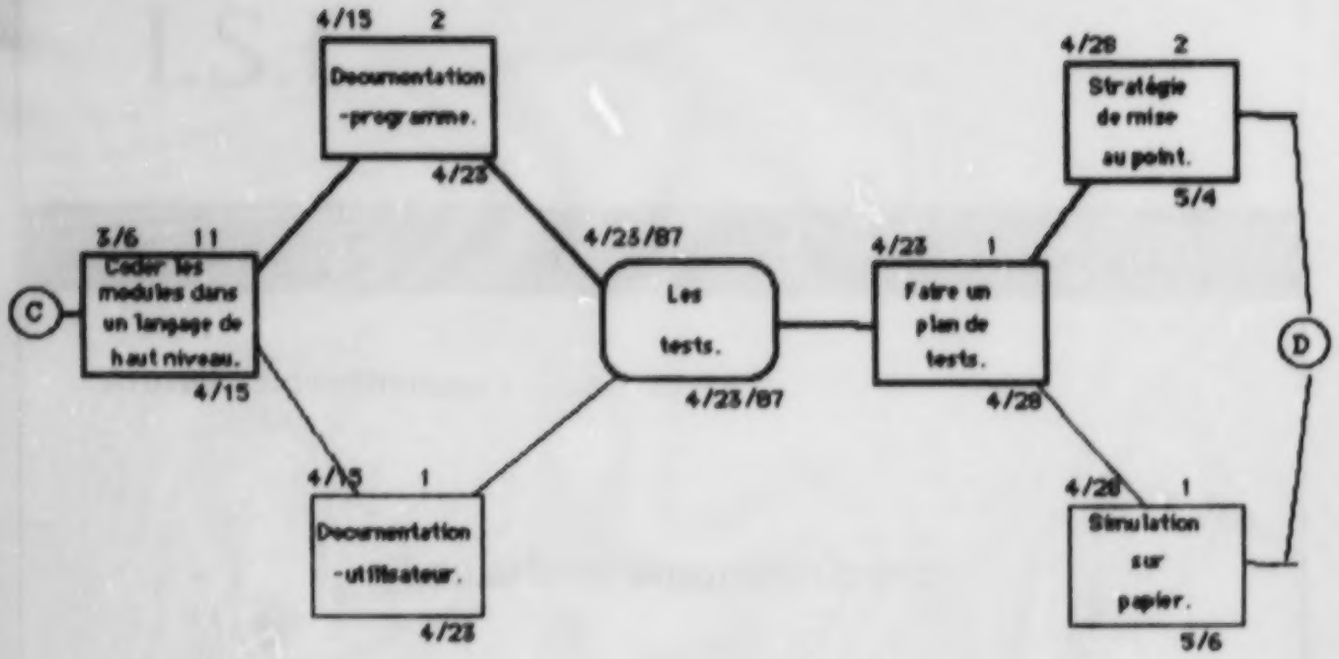
## La planification du projet-étudiant

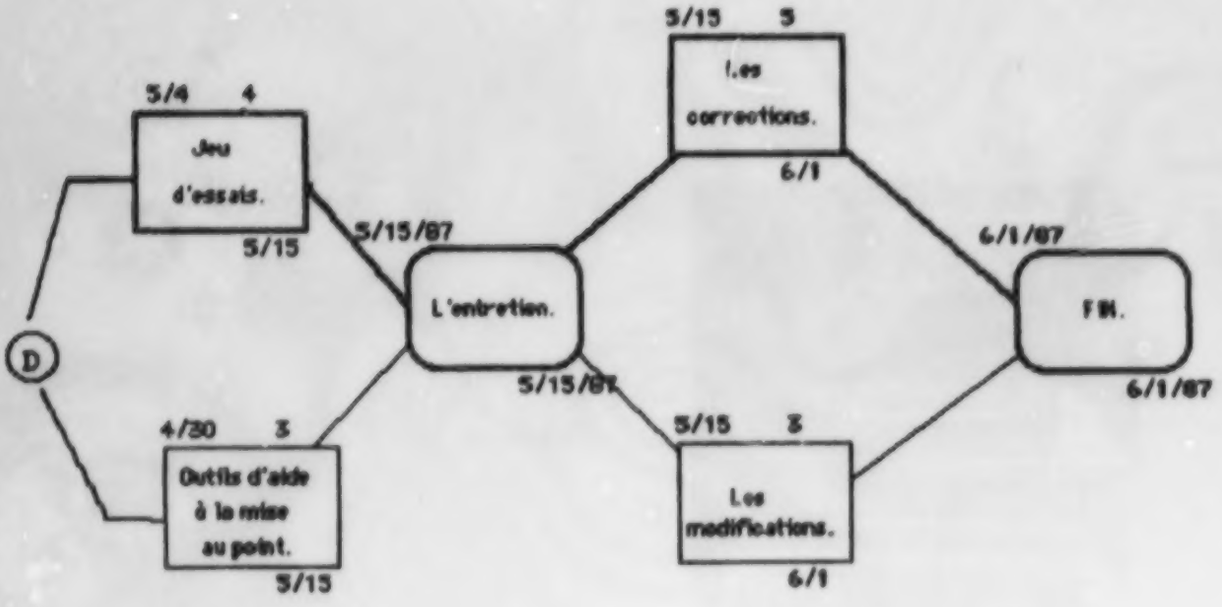
La planification du projet-étudiant doit tenir compte à la fois de l'horaire-étudiant et du calendrier scolaire de l'école. (horaire 6 ou 9 jours, congés statutaires, interétapes ...)




A
2/2 1
Faire une simulation écrite.
2/4
2/4/87
L'analyse du projet.
2/4/87
2/4 1
Définir les grandes étapes.
2/10
2/10 1
Faire l'arbre algorithmique.
2/16
2/10 2
Écrire le pseudo-langage
2/16
B


B
2/16/87
Les spécifications du projet.
2/16/87
2/16 1
Reformuler les grandes étapes.
2/18
2/18 2
Raffiner l'arbre algorithmique.
3/6
2/18 3
Raffiner les étapes du pseudo-langage
3/6
3/6/87
La réalisation du projet.
3/6/87
C

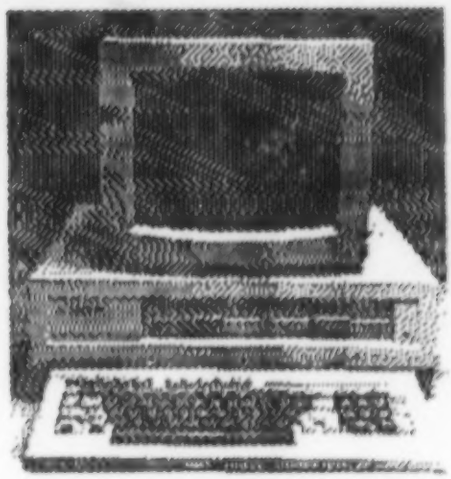

4/15 2
Documentation -programme.
4/23
4/28 2
Stratégie de mise au point.
5/4
3/6 11
Coder les modules dans un langage de haut niveau.
4/15
C
4/23/87
Les tests.
4/23/87
4/23 1
Faire un plan de tests.
4/28
D
4/15 1
Documentation -utilisateur.
4/23
4/28 1
Simulation sur papier.
5/6


5/15 3
Les corrections.
6/1
3/4 4
Jeu d'essais.
5/15
5/15/87
L'entretien.
5/15/87
6/1/87
FM.
6/1/87
D
4/30 3
Outils d'aide à la mise au point.
5/15
5/15 3
Les modifications.
6/1

# I.S.I.

Arbres algorithmiques.....p. 37-44

## Les arbres algorithmiques



## Un outil de résolution de problèmes

Un arbre algorithmique est un structurogramme qui permet de représenter graphiquement un algorithme. Cet outil est d'une grande utilité pour faire la décomposition du problème global en sous-problèmes, eux-mêmes décomposés en actions plus élémentaires jusqu'à ce que le niveau choisi pour la description du problème convienne au langage utilisé.

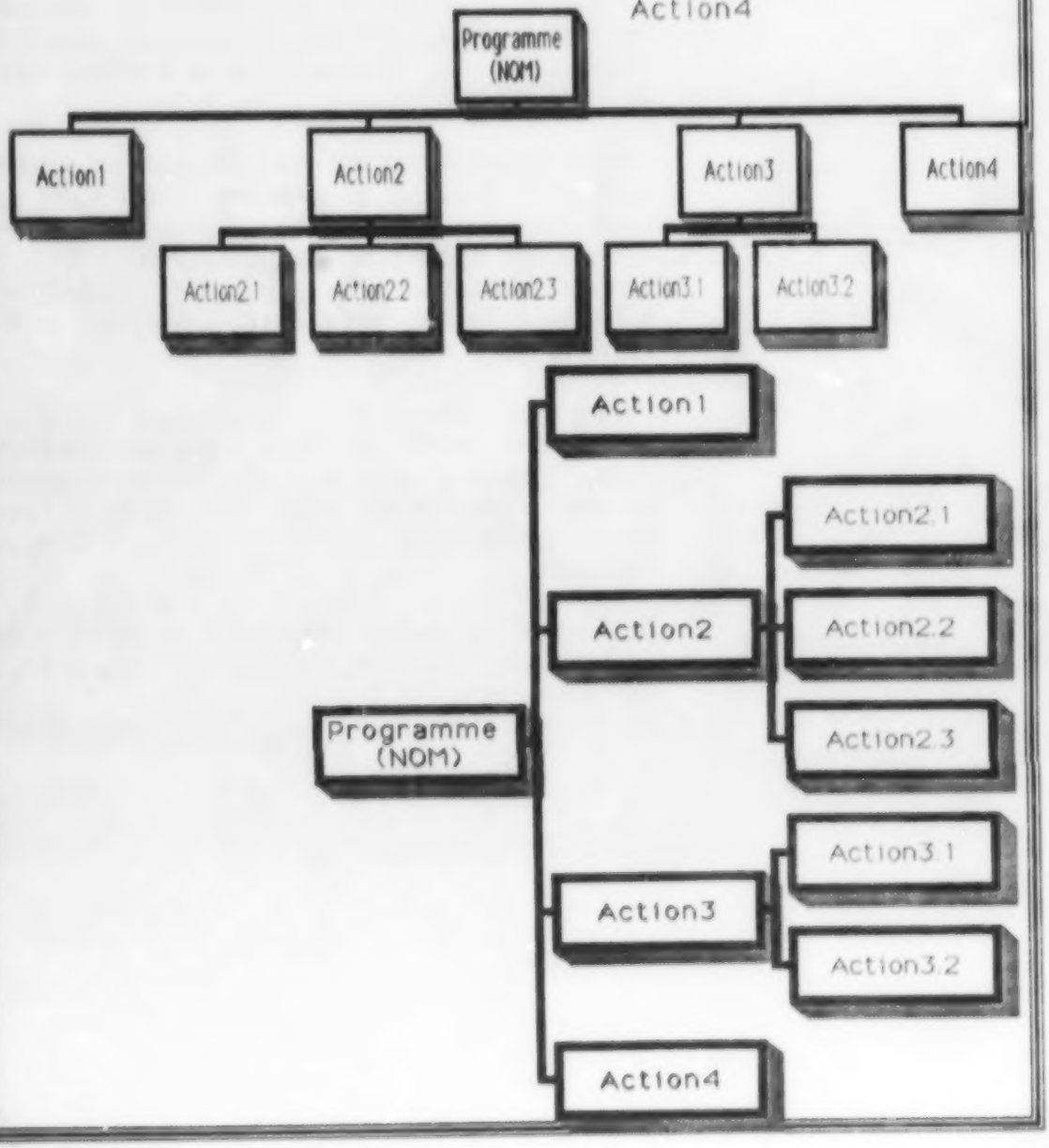
L'arbre algorithmique est non seulement un outil de planification et d'analyse du problème, il est aussi un instrument très riche de synthèse de fin de projet et d'évaluation critique de "toute l'opération". L'utilisateur doit être conscient que le moment d'utilisation de cet outil implique une méthodologie appropriée, sinon l'élève ne verra aucun sens à son usage.

L'arbre algorithmique se lit de gauche à droite et de haut en bas. Les modules de même niveau sont indépendants et chaque module situé à un niveau inférieur contribue à la réalisation du module situé immédiatement au-dessus. Cette représentation permet de découvrir la logique d'un algorithme.

Les pages qui suivent présentent les différents schémas informatiques fondamentaux sous la forme d'arbres algorithmiques.

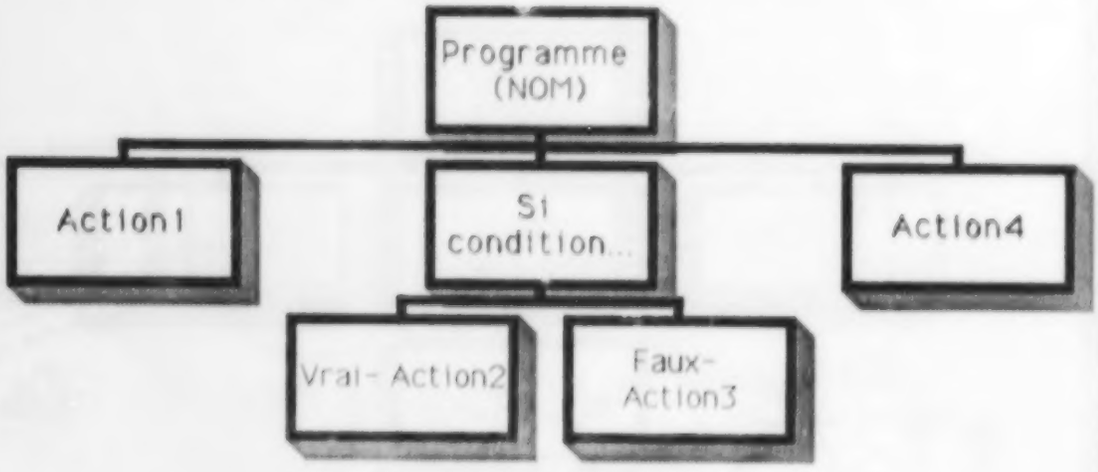
**La séquence.**

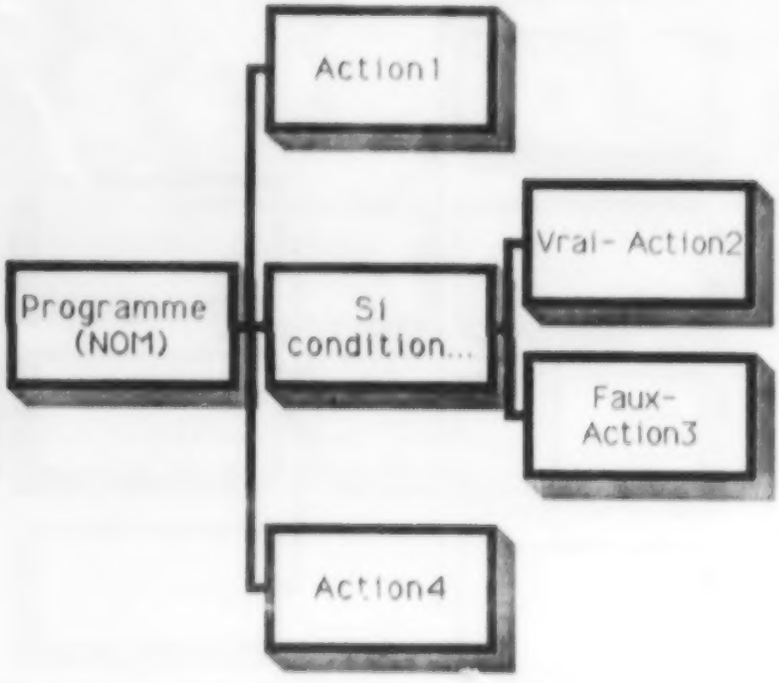
```
Programme (NOM)
  Action1
  Action2
     Action2.1
     Action2.2
     Action2.3
  Action3
     Action3.1
     Action3.2
  Action4
```



**Le branchement conditionnel.**

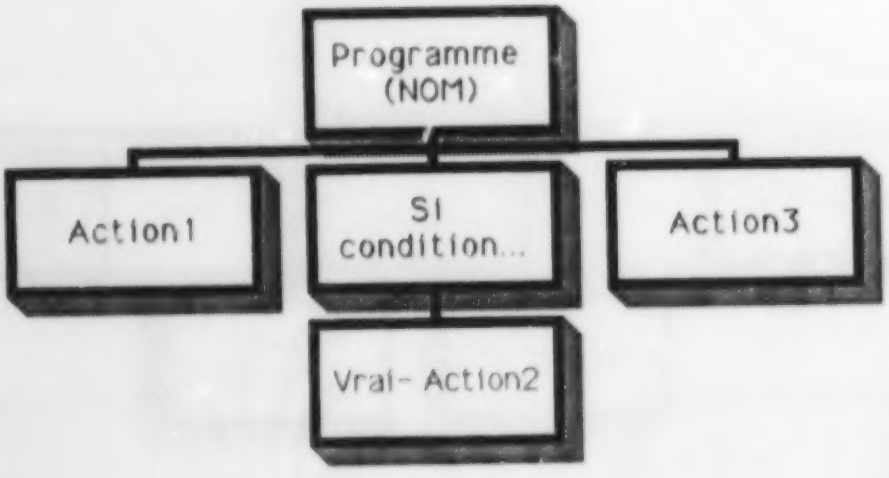
```
Programme (NOM)
  Action1
  SI condition...
     Vrai- Action2
     Faux- Action3
  FINSI
  Action4
```

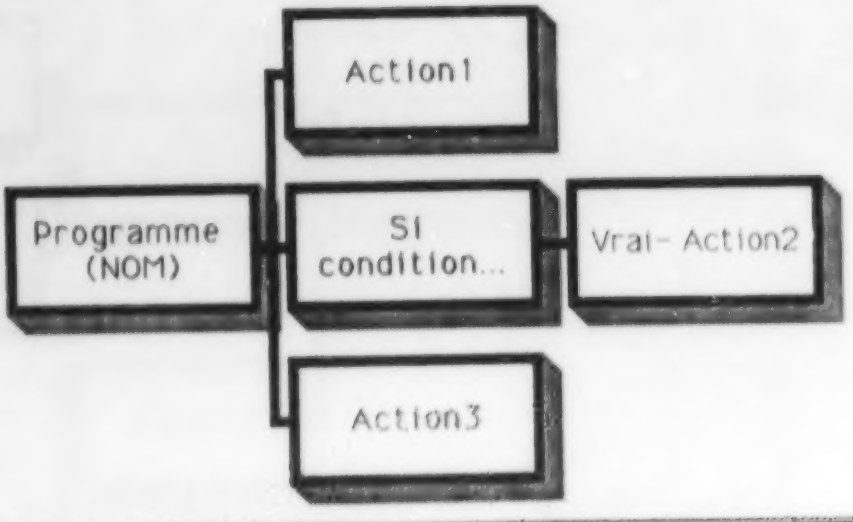


**Le branchement conditionnel.**

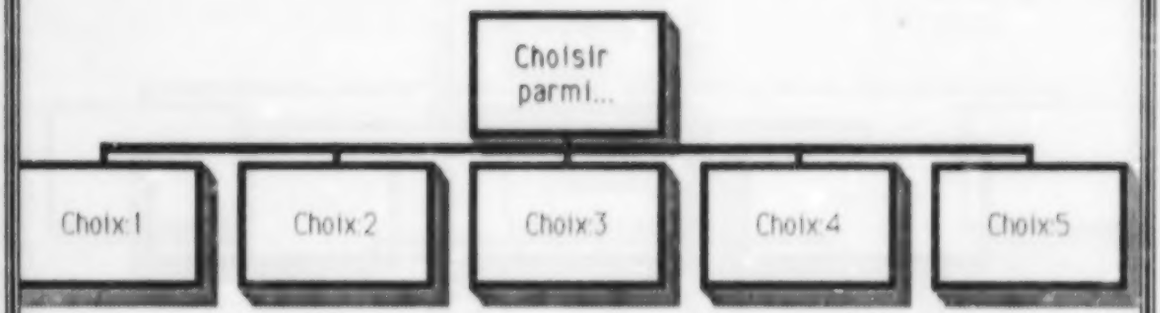
```
Programme (NOM)  Action1
  SI condition...
     Vrai- Action2
  FINSI
  Action3
```

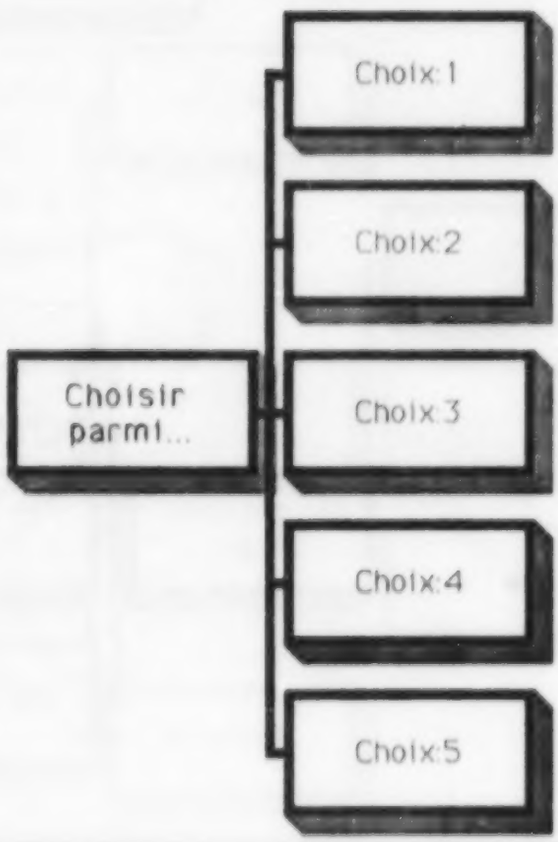


**Le choix parmi...**

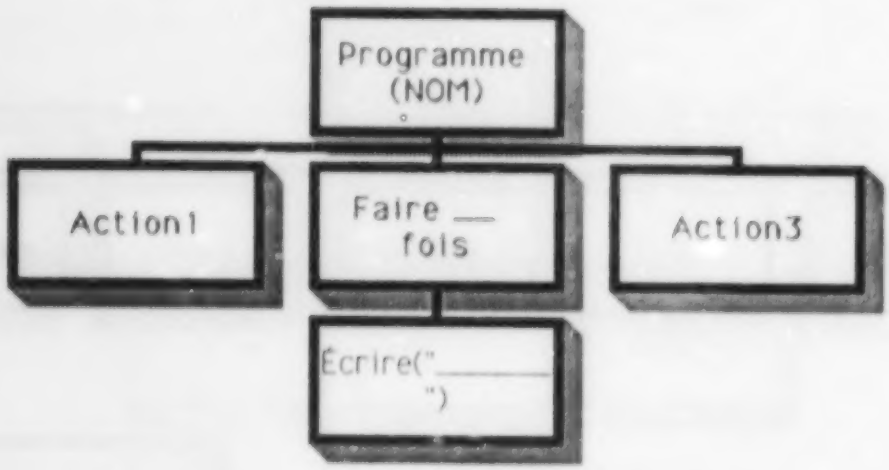
Choisir parmi... Choix:1
Choix:2
Choix:3
Choix:4
Choix:5

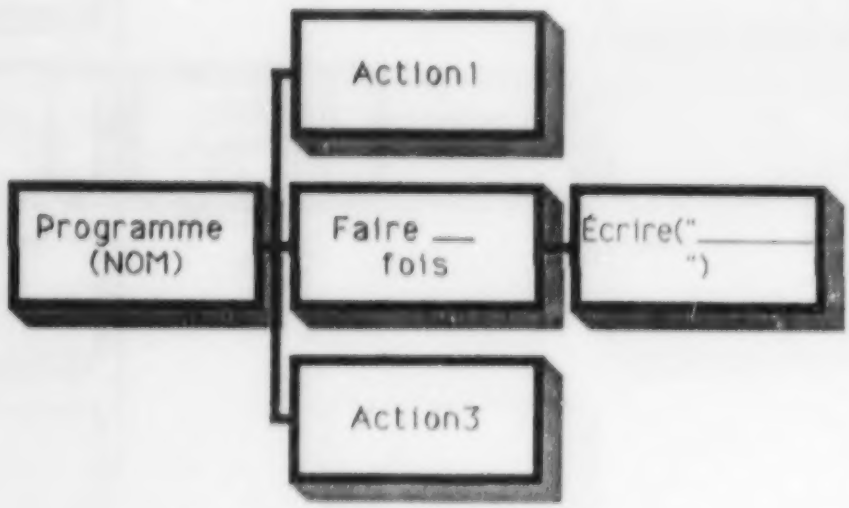


**La structure répétitive.**

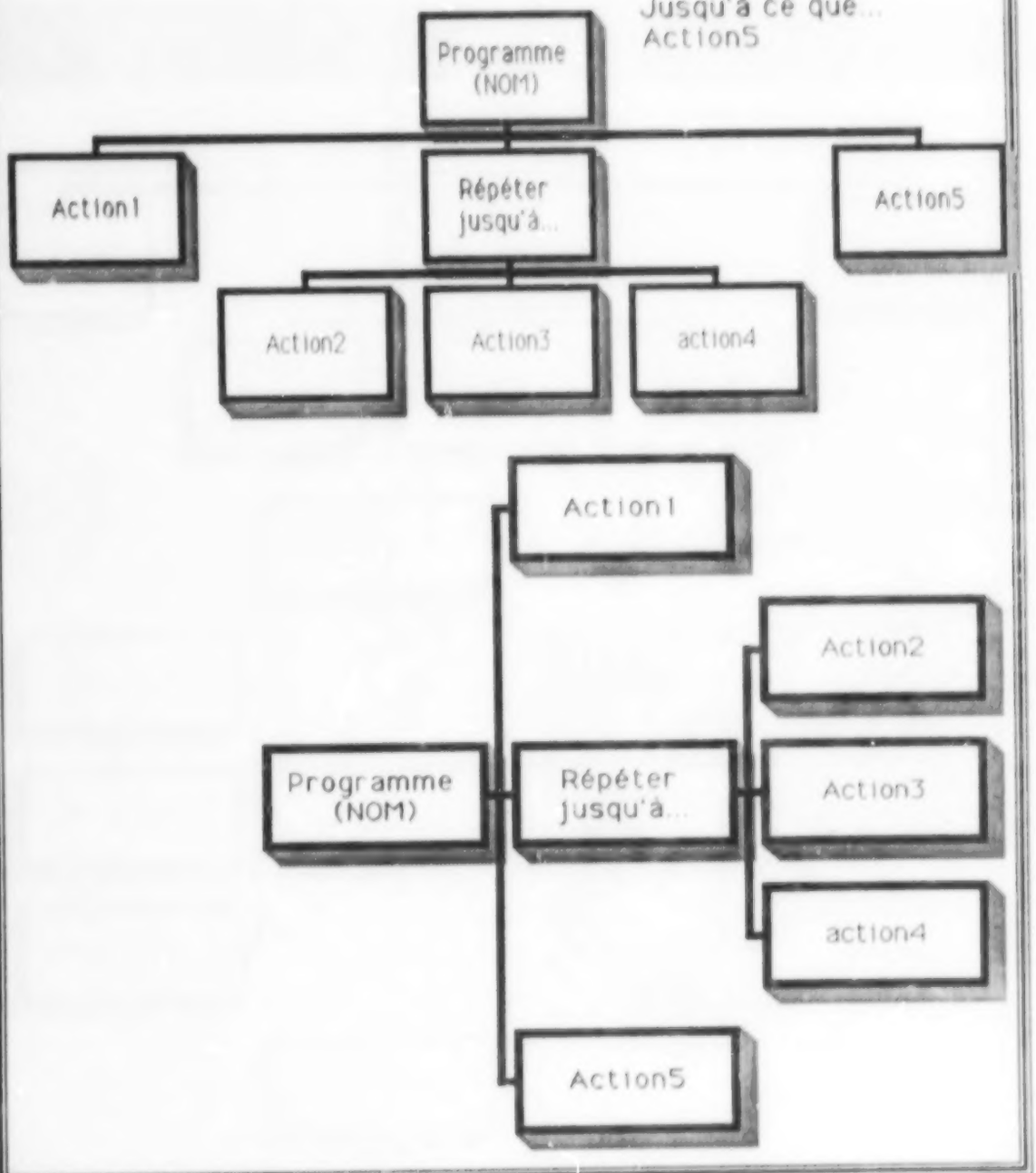
```
Programme (NOM) Action1
 BOUCLE dont variable...
    Écrire("_____")
 FIN DE BOUCLE
 Action3
```



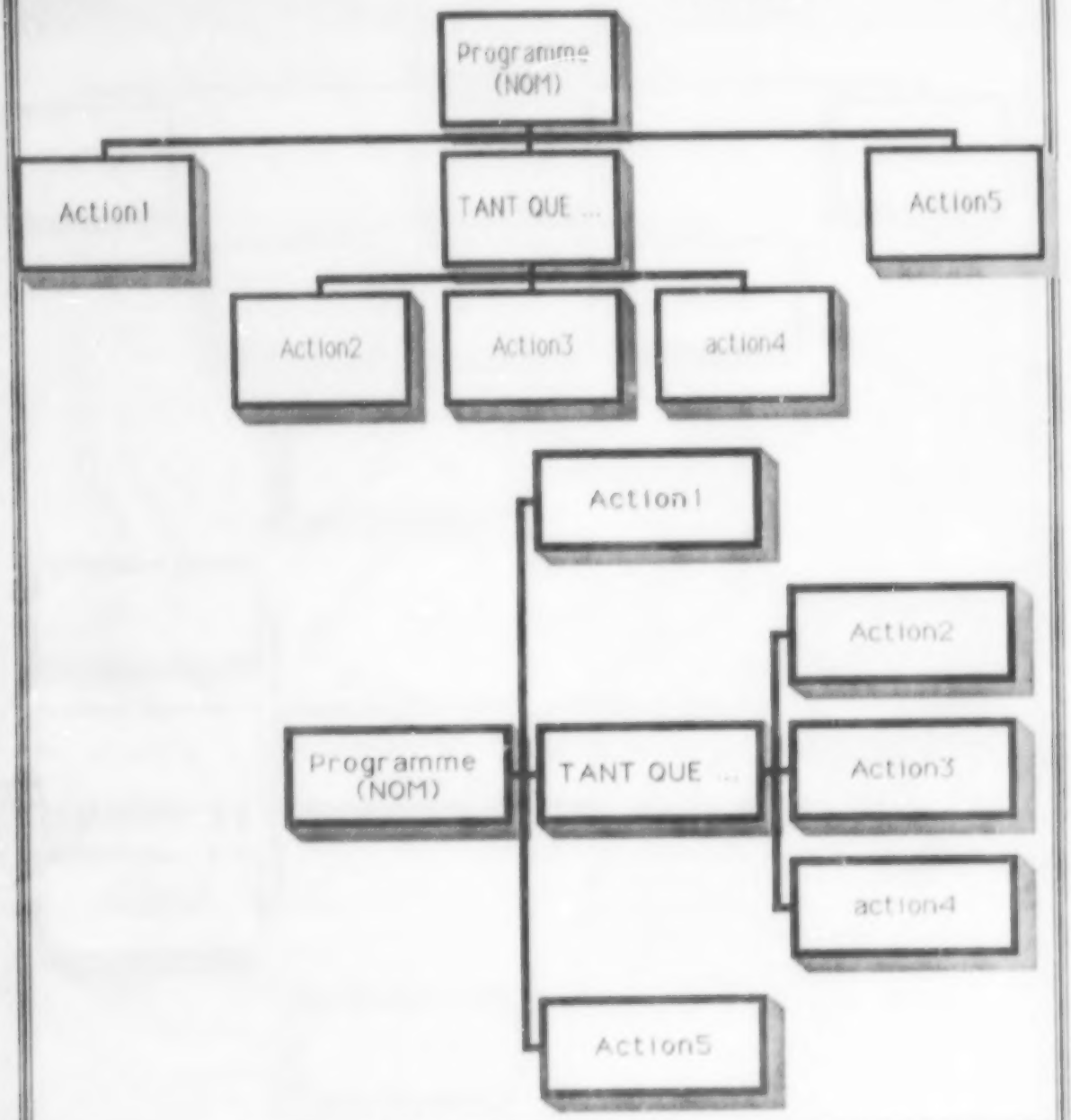
**La structure répétitive.**

```
Programme (NOM)
  Action1
  Répéter
      Action2
      Action3
      Action4
  Jusqu'à ce que...
  Action5
```



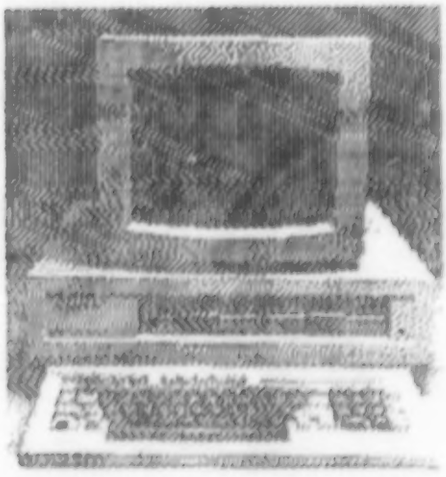
**La structure répétitive.**

```
Programme (NOM)  Action1
  TANT QUE ...
     Action2
     Action3
     Action4
  FINTANT
  Action5
```



# I.S.I.

Bibliographie.....p. 47



# BIBLIOGRAPHIE.


1. Pierre N. Robillard, Conception de logiciels. Ecole Polytechnique de Montréal, 1984.

2. Gouvernement du Québec, Direction des programmes. Introduction à la Science de l'informatique. Programme d'études secondaires, 1982. (Doc. no. 16-0081)

3. Gouvernement du Québec, D.G.D.P. Guide pédagogique du cours d'introduction à la Science de l'informatique. 1984. (Doc. no. 16-0081-01)

4. Michel Arcouet et Laurent Bélisle, Projets et Programmes. erpi 1986.

5. Alain Taurisson et André Petitguillaume, A vous de jouer , Modulo Editeur, 1983.

6. Claude Coulombe et Richard Young, La programmation structurée en BASIC, Turgeon 1986.

7. Jack Emmerichs, Apprendre à programmer, Modulo Editeur , 1985.

8. Mike James, Mieux programmer, Modulo Editeur, 1985.

9. Gouvernement du Québec, Direction générale des régions, Guide de formation pour un programme de 5 jours à l'intention des enseignants et des enseignantes du cours ISI, 1986. (Doc. no. 51-5067)

Gouvernement du Québec
Ministère
de l'Éducation

18-0081-03
6869-103